To set up a GitHub workflow with dev, QA, and prod branches, and manage permissions for developers, testers, team leads, and product owners, follow these steps:

**1. Branch Structure**

* **dev**: Main development branch where developers merge their feature branches.
* **QA**: Branch for testing. Testers will merge dev into QA for testing.
* **prod**: Production branch. Team lead/product owner will merge QA into prod for deployment.

**2. Permissions Setup**

* **Developers (4 persons)**:
  + Access to dev branch.
  + Can create feature branches from dev.
  + Can push to their feature branches.
  + Can create pull requests (PRs) from feature branches to dev.
  + Cannot directly push to dev or QA.
* **Testers (2 persons)**:
  + Access to QA branch.
  + Can create PRs from dev to QA.
  + Cannot directly push to QA or prod.
* **Team Lead/Product Owner**:
  + Access to prod branch.
  + Can merge QA into prod.
  + Can approve PRs from dev to QA and QA to prod.

**3. Steps for Collaboration**

**For Developers:**

1. **Create Feature Branches**:
   * Developers create feature branches from dev:

git checkout -b feature/feature-name dev

1. **Work on Feature Branches**:
   * Developers work on their feature branches and push changes:

git add .

git commit -m "Add feature XYZ"

git push origin feature/feature-name

1. **Create Pull Request (PR)**:
   * Developers create a PR from their feature branch to dev.
   * Team lead/reviewers review the PR and approve/merge it.
2. **Delete Feature Branch**:
   * After the PR is merged, the feature branch is deleted.

**For Testers:**

1. **Merge dev into QA**:
   * Testers create a PR from dev to QA for testing.
   * Team lead approves the PR and merges it.
2. **Test on QA**:
   * Testers test the changes in the QA branch.
   * If issues are found, they create issues/tickets for developers to fix.

**For Team Lead/Product Owner:**

1. **Merge QA into prod**:
   * After testing is complete, the team lead/product owner creates a PR from QA to prod.
   * Approve and merge the PR to deploy to production.

**4. GitHub Permissions Configuration**

1. **Repository Settings**:
   * Go to **Settings > Collaborators and teams**.
   * Add developers, testers, team lead, and product owner as collaborators.
2. **Branch Protection Rules**:
   * Go to **Settings > Branches**.
   * Add branch protection rules for dev, QA, and prod:
     + Require pull request reviews before merging.
     + Require status checks to pass before merging.
     + Restrict who can push to these branches:
       - dev: Only team lead can merge PRs.
       - QA: Only testers and team lead can merge PRs.
       - prod: Only team lead/product owner can merge PRs.
3. **Team Permissions**:
   * Create teams in GitHub:
     + **Developers Team**: Give write access to dev and feature branches.
     + **Testers Team**: Give write access to QA.
     + **Team Lead/Product Owner Team**: Give admin access to all branches.

**5. Workflow Summary**

1. Developers:
   * Create feature branches from dev.
   * Push changes to feature branches.
   * Create PRs to dev.
   * Delete feature branches after PR is merged.
2. Testers:
   * Create PRs from dev to QA.
   * Test changes in QA.
   * Report issues if any.
3. Team Lead/Product Owner:
   * Approve PRs from dev to QA.
   * Approve PRs from QA to prod.

**6. Example Commands**

* **Developer Workflow**:

git checkout dev

git pull origin dev

git checkout -b feature/new-feature

# Make changes

git add .

git commit -m "Add new feature"

git push origin feature/new-feature

# Create PR from feature/new-feature to dev

* **Tester Workflow**:

git checkout QA

git pull origin QA

git merge dev

git push origin QA

# Test changes in QA

* **Team Lead/Product Owner Workflow**:

git checkout prod

git pull origin prod

git merge QA

git push origin prod

This workflow ensures proper collaboration, code review, and testing before deploying to production. Adjust permissions and branch protection rules as needed for your team.